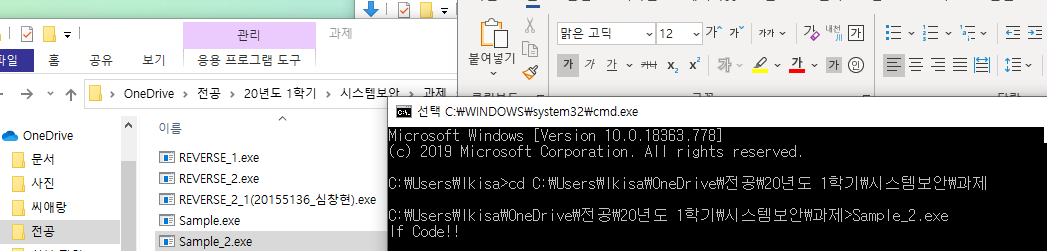
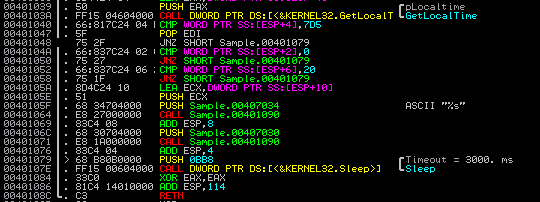
**문제 1. Sample.exe 파일 변경**

* **변경된 sample.exe 파일에 대한 스크린 샷 (변경된 sample.exe파일은 본 보고서와 함께 압축하여 제출함)**



* **Sample.exe 변경에 대한 이유 및 원리를 자유롭게 기술하시오.**



Sample.exe 파일을 이뮤니티 디버거로 실행했을 때 위와 같이 나타난다.

조건문은 00401040, 0040104A, 00401052 3개의 주소가 있다. 조건문이 참이면 ZF = 1이고 거짓이면 ZF = 0 이다. 또한 JNE 명령에서 ZF가 0이 아니면 00401079으로 점프를 하는데, 해당 주소를 살펴보면 3초 sleep 되면서 프로그램이 종료가 된다.

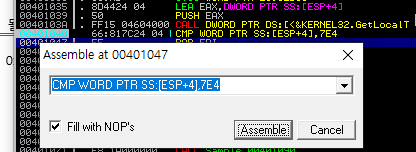
따라서 ZF값에 따라 조건문의 흐름이 달라진다는 것을 볼 수 있고, 조건문을 모두 참으로 만들어야 한다.

첫번째 조건문인 00401040부터 살펴보면

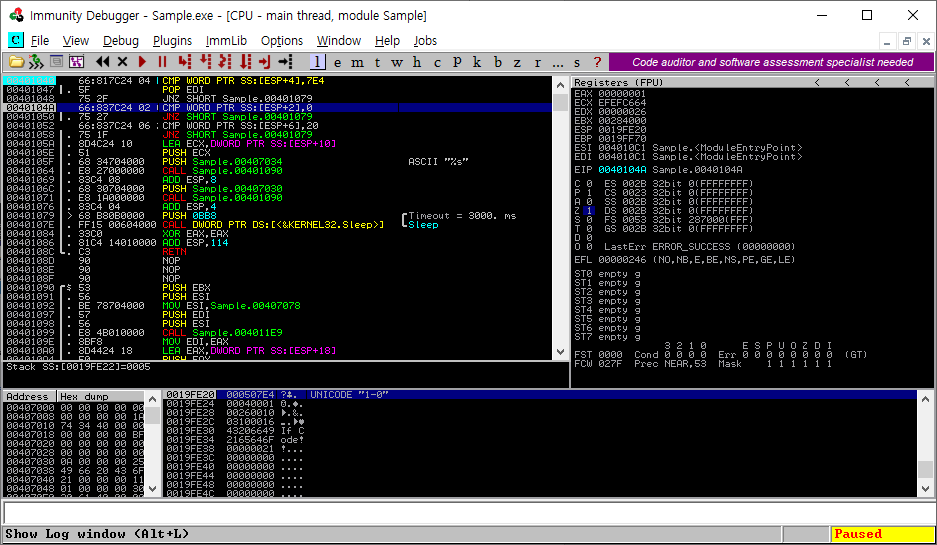


ESP+4 주소 2바이트와 0x07D5를 비교하는 명령이다. 값이 같으면 ZF에 1, 다르면 0을 넣는 명령이다.

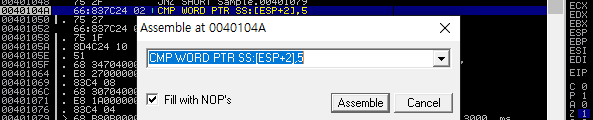
이 명령에서 실행 될 ZF에 의해 아래 00401048 주소의 점프 명령어가 결정이 된다.



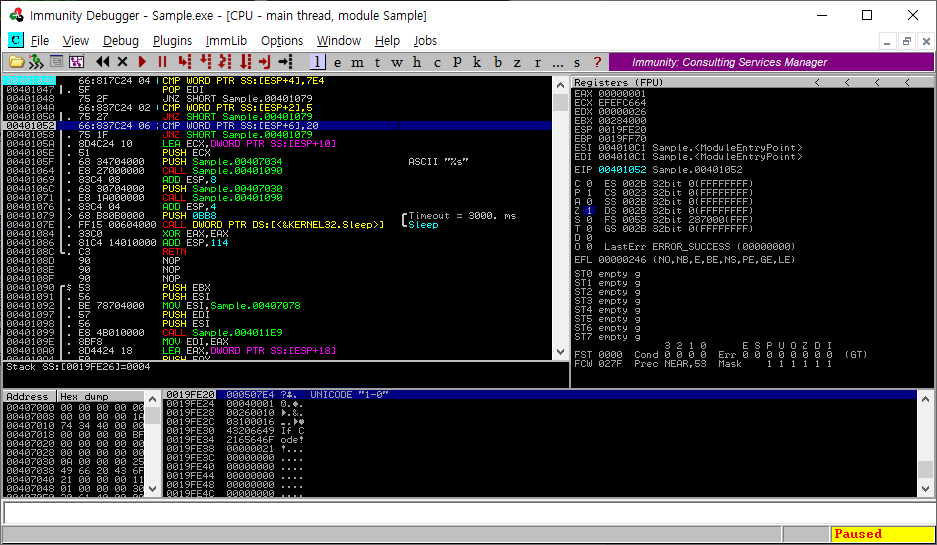
따라서 0x07D5를 0x07E4 으로 수정해주면 ZF값이 1이 되므로 00401048 주소의 점프문 다음 주소를 실행시킨다.



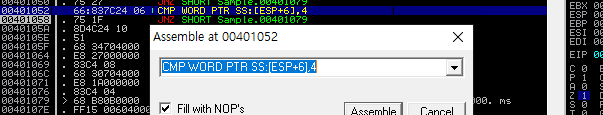
0040104A 주소의 명령은 ESP+2의 주소 2바이트와 0을 비교한다. 하단 변수값 출력 창을 보면 스택은 0005 이므로



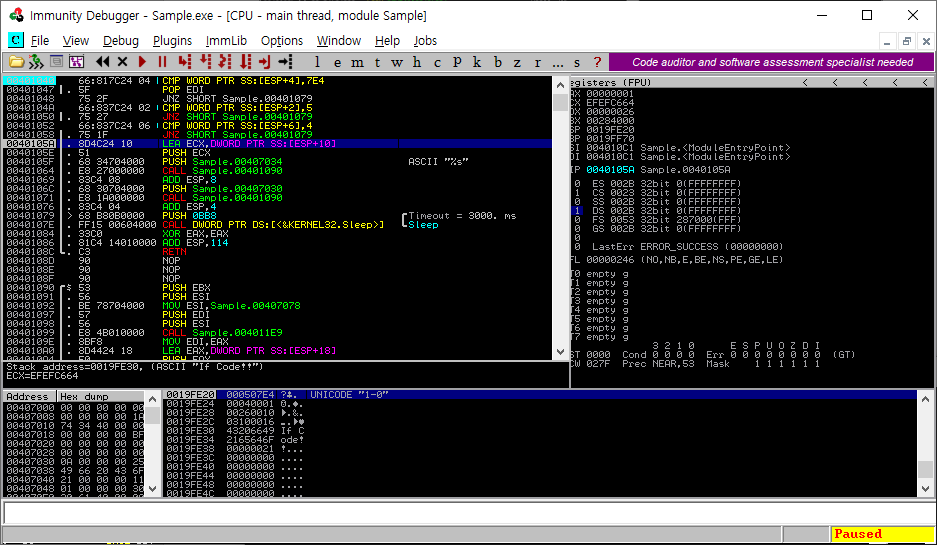
0에서 5으로 수정을 해주면 위와 동일하게 ZF값이 1이 되므로 00401050 점프문 다음 주소를 실행시킨다.



00401052 주소의 명령은 ESP+6의 주소 2바이트와 20을 비교한다. 이 조건도 마찬가지로 ZF 값을 1으로 설정해야 하므로 하단 스택의 주소와 동일하게 명령을 바꿔준다.

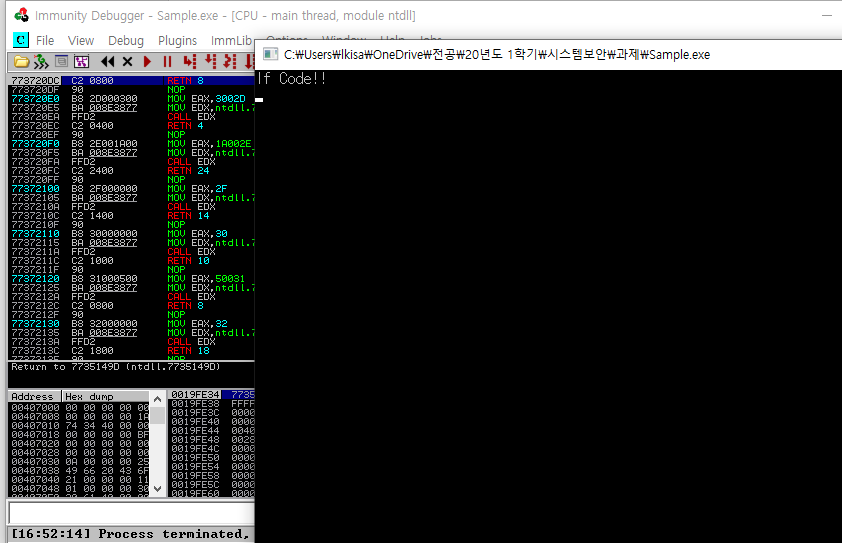


따라서 조건문의 ZF값을 모두 1으로 설정해주었기 때문에 JNE 명령어의 다음 명령을 실행할 수 있게 되었으므로 3초 sleep하는 주소로 점프를 하지 않는다.



정상적으로 ECX 레지스터에 “If Code!!” 문자열이 삽입되었고, 다음 명령어인 PUSH ECX를 통해 스택에 정상적으로 값이 들어간 것을 볼 수 있다.

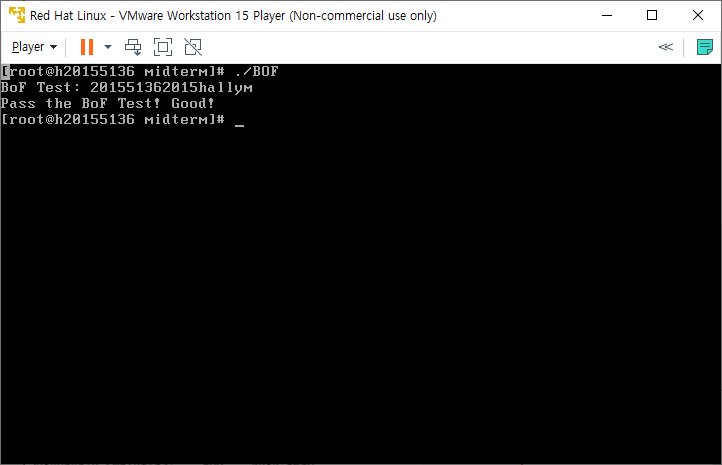
프로그램을 브레이크 포인트 없이 계속 실행을 시켜보면



If Code!!가 정상적으로 출력되는 것을 확인할 수 있다.

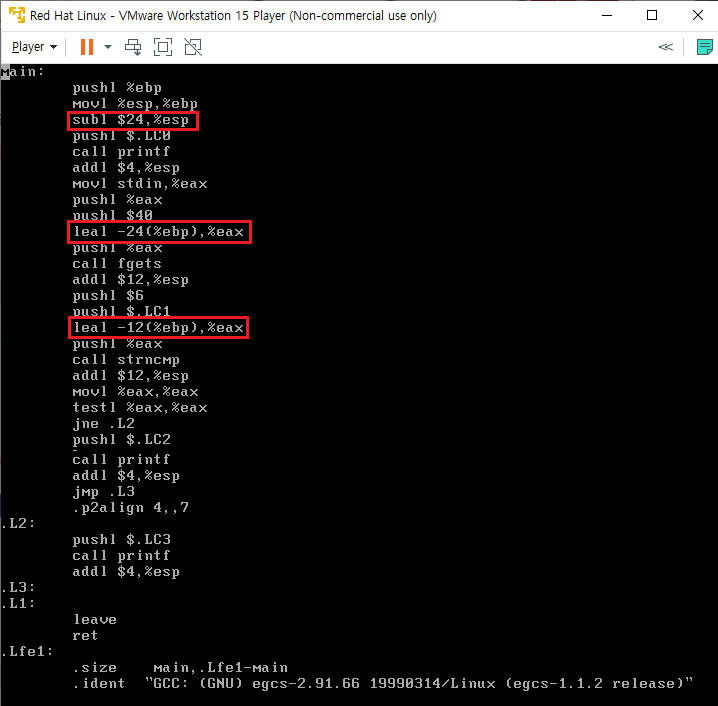
**문제 2. BOF약점 테스트**

* **Input값 조작으로 출력된 “Pass the BoF test! Good!” 스크린 샷**



* **BOF 취약점이 발생한 이유 및 BOF 취약점을 야기시키는 input 생성 원리에 대해 자유롭게 기술하시오.**

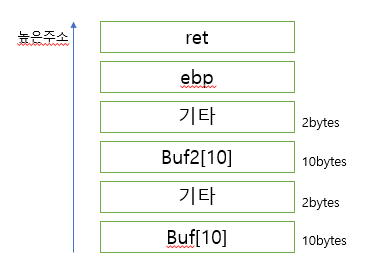
어셈블리로 컴파일 하여 열어보면 아래와 같은 어셈블리어 코드들이 나온다.



Subl $24, %esp 명령어는 스택에 24바이트 만큼의 용량을 할당해준다.

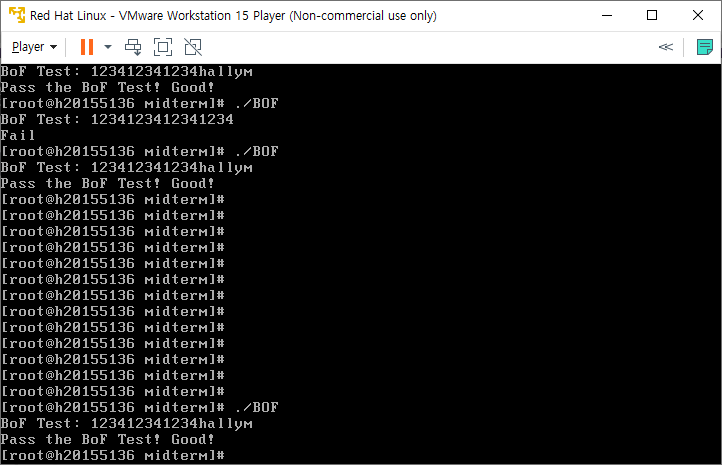
Fgets() 함수로 확인한 buf 위치는 epb-24 이고, strncmp로 확인한 buf2 위치는 epb-12 이다.

그림으로 나타내면 아래와 같이 나타낼 수 있다.



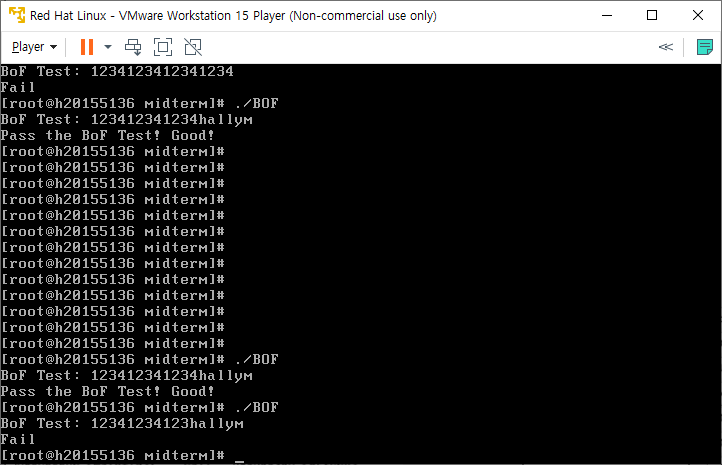
소스에서는 buf2에 “hallym”을 비교하는 명령이 있으므로 buf부터 12바이트를 쓰레기값으로 채운 뒤 hallym을 입력하면 Pass the BoF test! Good! 이 프린트 될 것이다.

입력에 “123412341234hallym”을 입력해보면



정상적으로 buf2값에 hallym이 덮어 씌워져서 조건문이 참인 모습을 볼 수 있다.

만약 11바이트를 쓰레기값으로 채운 뒤 hallym을 입력한다면



조건문이 거짓으로 되어 Fail이 뜬다.

즉 할당된 크기를 넘어 입력이 가능해질 경우 다른 할당된 메모리에 접근할 수 있다. 이 공격에서는 10바이트인 buf와 더미값인 2바이트의 합 12바이트를 넘어 문자열을 입력하는 경우에 buf2에 접근할 수 있었고 buf2 값을 조작하여 조건문을 통과할 수 있었다.